

[illegible][illegible]

```

TTTTTTTTT1  RRRRRRRR  NN      NN      SSSSSSSS
TTTTTTTTTT  RRRRRRRR  NN      NN      SSSSSSSS
      TT      RR      NN      NN      SS
      TT      RR      NN      NN      SS
      TT      RR      NNNN     NN      SS
      TT      RR      NNNN     NN      SS
      TT      RRRRRRRR  NN  NN  NN      SSSSSS
      TT      RRRRRRRR  NN  NN  NN      SSSSSS
      TT      RR  RR      NN      NNNN     SS
      TT      RR  RR      NN      NNNN     SS
      TT      RR      RR  NN      NN      SS
      TT      RR      RR  NN      NN      SS
      TT      RR      RR  NN      NN      SS
      TT      RR      RR  NN      NN      SS
      TT      RR      RR  NN      NN      SSSSSSSS
      TT      RR      RR  NN      NN      SSSSSSSS
      -----
      -----
BBBBBBBBBB  BBBB      BB      IIIIII  TTTTTTTTTT  SSSSSSSS
BBBBBBBBBB  BBBB      BB      IIIIII  TTTTTTTTTT  SSSSSSSS
      BB      BB      II      TT      SS
      BB      BB      II      TT      SS
      BB      BB      II      TT      SS
      BB      BB      II      TT      SS
      BBBB      BB      II      TT      SSSSSS
      BBBB      BB      II      TT      SSSSSS
      BB      BB      II      TT      SS
      BB      BB      II      TT      SS
      BB      BB      II      TT      SS
      BB      BB      II      TT      SS
      BBBB      BB      IIIIII  TT      SSSSSSSS
      BBBB      BB      IIIIII  TT      SSSSSSSS
      ...
      ...
      ...
      ...

```


E 8
15-Sep-1984 23:55:21
14-Sep-1984 12:28:13

VAX-11 Bliss-32 V4.0-742
[ERF.SRC]TRNS_BITS.B32;1

Page 1
(1)

```
0001 0 MODULE TRANSLATE BITS
0002 0 (%TITLE 'Translate bits to text'
0003 0 Ident = 'V04-000') =
0004 1 Begin
0005 1
0006 1
0007 1 *****
0008 1 *
0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
0011 1 * ALL RIGHTS RESERVED. *
0012 1 *
0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
0018 1 * TRANSFERRED. *
0019 1 *
0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
0022 1 * CORPORATION. *
0023 1 *
0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
0026 1 *
0027 1 *
0028 1 *****
0029 1
0030 1 ++
0031 1 FACILITY: ERF, Error Log Report Generator
0032 1
0033 1 ABSTRACT:
0034 1
0035 1 This module contains that translates bit to text for single
0036 1 bit fields.
0037 1
0038 1 ENVIRONMENT:
0039 1
0040 1 VAX/VMS operating system, user mode.
0041 1
0042 1 AUTHOR: Sharon Reynolds, CREATION DATE: 20-Feb-1984
0043 1
0044 1
0045 1 V04-001 SAR0284 Sharon A. Reynolds 2-Jul-1984
0046 1 - Added FA0 string building as a function of this
0047 1 routine.
0048 1 --
0049 1 REQUIRE 'SRC$:ERFDEF.REQ' ;
0050 1
0051 1 FORWARD ROUTINE
0052 1 Translate_bits ;
0053 1
0054 1 EXTERNAL ROUTINE
0055 1 Lib$get_vm ;
0056 1
0057 1 OWN
```

Page 2
(1)

```

: 58      0343 1      Arglist: vector [31],
: 59      0344 1      Fao_string_desc: $BBLOCK [8],
: 60      0345 1      Initial_execution: Initial (0);
: 61      0346 1

```

[illegible]


```

63 0347 1 GLOBAL Routine TRANSLATE_BITS (register_contents,register_mask,desc_table,
64 0348 1 out_arglist,fao_string,control_flag) =
65 0349 2 Begin
66 0350 2
67 0351 2
68 0352 2
69 0353 2
70 0354 2
71 0355 2
72 0356 2
73 0357 2
74 0358 2
75 0359 2
76 0360 2
77 0361 2
78 0362 2
79 0363 2
80 0364 2
81 0365 2
82 0366 2
83 0367 2
84 0368 2
85 0369 2
86 0370 2
87 0371 2
88 0372 2
89 0373 2
90 0374 2
91 0375 2
92 0376 2
93 0377 2
94 0378 2
95 0379 2
96 0380 2
97 0381 2
98 0382 2
99 0383 2
100 0384 2
101 0385 2
102 0386 2
103 0387 2
104 0388 2
105 0389 2
106 0390 2
107 0391 2
108 0392 2
109 0393 2
110 0394 2
111 0395 2
112 0396 2
113 0397 2
114 0398 2
115 0399 2
116 0400 2
117 0401 2
118 0402 2
119 0403 2

```

Functional Description:

Given the mask of bits to translate, this routine will translate bit to text for a given register. It will build a list of the addresses of the descriptors of the text that needs to be output. If the 'control_flag' is '0' then all set bits are translated; if the 'control_flag' is '1' then all clear bits are translated. An FAO string will be built and the address returned to the calling routine.

Calling Sequence:

TRANSLATE_BITS (register_contents,register_mask,descriptor_table, out_arglist,fao_string,control_flag) ;

Input Parameters:

register_contents = binary data,
register_mask = longword indicating which bits should be translated (bit set = bit translated),
descriptor_table = address of descriptor table pointing to text,
control_flag = if '0' then translate bits that are set, if '1' then translate bits that are clear

Output Parameters:

out_arglist = returns the address of a list of the addresses of the descriptors for the text that should be output.
fao_string = returns the address of descriptor for the FAO string that got built.

--

LOCAL

Blk_index: Initial (0),
Index: Initial (0),
Arg_index: Byte
Initial (0),
End_bit: Byte
Initial (byte(31)),
Start_bit: Byte
Initial (0),
Bit_position: Initial (0),
Field_size: Byte,
Start_position,
Status ;

MAP

Register_contents: REF \$BBLOCK,
Register_mask: REF \$BBLOCK,
Desc_table: REF BLOCKVECTOR [,2,long] ;


```

120 0404 2
121 0405 2 BUILTIN
122 0406 2 FFS ;
123 0407 2
124 0408 2
125 0409 2 If first execution, then get the fao string storage.
126 0410 2
127 0411 2 If .initial_execution EQL 0
128 0412 2 Then
129 0413 2 Begin
130 0414 2 Initial_execution = 1 ;
131 0415 2 If NOT (status = LIB$GET_VM (%REF(500),fao_string_desc[dsc$a_pointer]))
132 0416 2 Then
133 0417 2 Signal_stop (.status) ;
134 0418 2 End ;
135 0419 2
136 0420 2
137 0421 2 Find starting bit for the translation (from the mask).
138 0422 2
139 0423 2 Field_size = 32 ;
140 0424 2 Start_position = 0 ;
141 0425 2
142 0426 2 Until .start_position GEQ 32 do
143 0427 2 Begin
144 0428 2 If NOT (FFS (start_position,field_size,
145 0429 2 .register_mask,bit_position))
146 0430 2 Then
147 0431 2 Begin
148 0432 2 If .start_position EQL 0
149 0433 2 Then
150 0434 2 Start_bit = .bit_position
151 0435 2 Else
152 0436 2 End_bit = .bit_position ;
153 0437 2 End ;
154 0438 2
155 0439 2 Start_position = .bit_position + 1 ;
156 0440 2 Field_size = 32 - .start_position ;
157 0441 2 End ;
158 0442 2
159 0443 2
160 0444 2 Determine if the bits should be translated. If so, determine if the
161 0445 2 bit is set/clear and whether it should be translated. If so, get the
162 0446 2 address of the descriptor for the associated text.
163 0447 2
164 0448 2 Incr index from .start_bit to .end_bit do
165 0449 2 Begin
166 0450 2 If .register_mask[0,.index,1,0]
167 0451 2 Then
168 0452 2 Begin
169 0453 2 If (NOT ..control_flag AND .register_contents[0,.index,1,0])
170 0454 2 OR ((..control_flag) AND (NOT .register_contents[0,.index,1,0]))
171 0455 2 Then
172 0456 2 Begin
173 0457 2 Arglist[.arg_index] = desc_table[.blk_index,0,0,0,0] ;
174 0458 2 Arg_index = .arg_index + 1 ;
175 0459 2 End ;
176 0460 2

```



```

: 177      0461      4      Blk_index = .blk_index + 1 ;
: 178      0462      3      End ;
: 179      0463      2      End ;
: 180      0464      1      -----
: 181      0465      0      Determine whether any bits were translated, set up the output argument
: 182      0466      0      address and the address of the descriptor for the FAO string for return
: 183      0467      0      to the calling routine, build and FAO string, and return with a true value.
: 184      0468      0
: 185      0469      0
: 186      0470      0      if .arg_index NEQ 0
: 187      0471      0      Then
: 188      0472      0          Begin
: 189      0473      0              .Out_arglist = arglist ;
: 190      0474      0              .fao_string = fao_string_desc ;
: 191      0475      0
: 192      0476      0              fao_string_desc[dsc$w.length] = 0 ;
: 193      P 0477      0              BUILD_FAO_STRING (1, '!39< !>!AS', fao_string_desc,
: 194      0478      0                  (.arg_index - 1), '!39< !>!AS', fao_string_desc) ;
: 195      0479      0              Return true ;
: 196      0480      0              End ;
: 197      0481      0
: 198      0482      0      Return false ;
: 199      0483      0
: 200      0484      1      End ; ! Routine

```

```

                                .TITLE TRANSLATE_BITS Translate bits to text
                                .IDENT  \V04-000\
                                .PSECT $PLIT,NOWRT,NOEXE, PIC,2

00 00 53 41 21 3E 21 20 3C 39 33 21 00000 P.AAB: .ASCII  \!39< !>!AS\<0><0>
                                010E000A 0000C P.AAA: .LONG   17694730
                                00000000' 00010 .ADDRESS P.AAB
53 41 21 3E 21 20 3C 39 33 21 2F 21 00014 P.AAD: .ASCII  \!39< !>!AS\
                                010E000C 00020 P.AAC: .LONG   17694732
                                00000000' 00024 .ADDRESS P.AAD
                                .PSECT $OWN$,NOEXE, PIC,2

                                00000 ARGLIST:.BLKB 124
                                0007C FAO_STRING_DESC:
                                    .BLKB 8
                                00000000 00084 INITIAL_EXECUTION:
                                    .LONG 0
                                STR_DESC= P.AAA
                                STR_DESC= P.AAC
                                .EXTRN LIB$GET_VM
                                .PSECT $CODE,NOWRT, PIC,2

                                OFFC 00000 .ENTRY TRANSLATE_BITS, Save R2,R3,R4,R5,R6,R7,R8,- ; 0347
                                5B 00000000' 00 9E 00002 R9,R10,R11
                                5A 00000000' 00 9E 00009 STR_DESC, R11
                                5E 04 C2 00010 MOVAB FAO_STRING_DESC, R10
                                SUBL2 #4,-SP

```


				50	D4	00013	CLRL	INDEX	0349
				59	94	00015	CLRB	ARG_INDEX	
		55		1F	90	00017	MOVB	#31, END_BIT	
				54	94	0001A	CLRB	START_BIT	
				52	7C	0001C	CLRQ	BLK_INDEX	
			08	AA	D5	0001E	TSTL	INITIAL_EXECUTION	0411
				23	12	00021	BNEQ	1\$	
		08	AA	01	D0	00023	MOVL	#1, INITIAL_EXECUTION	0414
				04	AA	9F	PUSHAB	FAO_STRING_DESC+4	0415
		04	AE	8F	3C	0002A	MOVZWL	#500, 4(SP)	
				04	AE	9F	PUSHAB	4(SP)	
		00000000G	00	02	FB	00033	CALLS	#2, LIB\$GET_VM	
			09	50	E8	0003A	BLBS	STATUS, 1\$	
				50	DD	0003D	PUSHL	STATUS	0417
		00000000G	00	01	FB	0003F	CALLS	#1, LIB\$STOP	
			51	20	90	00046	MOVB	#32, FIELD_SIZE	0423
				50	D4	00049	CLRL	START_POSITION	0424
			20	50	D1	0004B	CMPL	START_POSITION, #32	0426
				1E	18	0004E	BGEQ	5\$	
53	08	BC	51	50	EA	00050	FFS	START_POSITION, FIELD_SIZE, -	0428
								@REGISTER_MASK, BIT_POSITION	
				0C	13	00056	BEQL	4\$	
				50	D5	00058	TSTL	START_POSITION	0432
				05	12	0005A	BNEQ	3\$	
			54	53	90	0005C	MOVB	BIT_POSITION, START_BIT	0434
				03	11	0005F	BRB	4\$	
			55	53	90	00061	MOVB	BIT_POSITION, END_BIT	0436
			50	A3	9E	00064	MOVAB	1(R3), START_POSITION	0439
		51	20	50	83	00068	SUBB3	START_POSITION, #32, FIELD_SIZE	0440
				DD	11	0006C	BRB	2\$	0426
			53	55	9A	0006E	MOVZBL	END_BIT, R3	0448
			51	54	9A	00071	MOVZBL	START_BIT, INDEX	
				51	D7	00074	DECL	INDEX	
				25	11	00076	BRB	10\$	
		20	08	51	E1	00078	BBC	INDEX, @REGISTER_MASK, 10\$	0450
			09	BC	E8	0007D	BLBS	@CONTROL_FLAG, 7\$	0453
				51	E0	00081	BBS	INDEX, @REGISTER_CONTENTS, 8\$	
				18	BC	E9	BLBC	@CONTROL_FLAG, 9\$	0454
			0C	51	E0	0008A	BBS	INDEX, @REGISTER_CONTENTS, 9\$	
				59	9A	0008F	MOVZBL	ARG_INDEX, R0	0457
		84	AA40	0C	BC42	7E	MOVAQ	@DESC_TABLE[BLK_INDEX], ARGLIST[R0]	
				59	96	00099	INCB	ARG_INDEX	0458
				52	D6	0009B	INCL	BLK_INDEX	0461
			D7	53	F3	0009D	AOBLEQ	R3, INDEX, 6\$	0448
				59	95	000A1	TSTB	ARG_INDEX	0470
				4C	13	000A3	BEQL	14\$	
				84	AA	9E	MOVAB	ARGLIST, @OUT_ARGLIST	0473
					6A	9E	MOVAB	FAO_STRING_DESC, @FAO_STRING	0474
					6A	B4	CLRW	FAO_STRING_DESC	0476
					6B	3C	MOVZWL	STR_DESC, R7	0478
				04	AB	D0	MOVL	STR_DESC+4, R8	
					01	D0	MOVL	#1, I	
					6A	3C	MOVZWL	FAO_STRING_DESC, R0	
				04	AA	C0	ADDL2	FAO_STRING_DESC+4, R0	
					57	28	MOV3	R7, (R8), (R0)	
60			68	57	A0	000C5	ADDW2	R7, FAO_STRING_DESC	
					01	F3	AOBLEQ	#1, I, T1\$	
EE			56						

	59		59	9A	000CC	MOVZBL	ARG_INDEX, R9
	57	14	AB	3C	000CF	MOVZWL	STR_DESC, R7
	58	18	AB	D0	000D3	MOVL	STR_DESC+4, R8
			56	D4	000D7	CLRL	I
			0E	11	000D9	BRB	13\$
	50		6A	3C	000DB	MOVZWL	FAO_STRING_DESC, R0
	50	04	AA	C0	000DE	ADDL2	FAO_STRING_DESC+4, R0
60	68		57	28	000E2	MOVCL	R7, (R8), (R0)
	6A		57	A0	000E6	ADDW2	R7, FAO_STRING_DESC
EE	56		59	F2	000E9	AOBLSS	R9, I, T2\$
	50		01	D0	000ED	MOVL	#1, R0
				04	000F0	RET	
			50	D4	000F1	CLRL	R0
			04	000F3	RET		

0479
0482
0484

; Routine Size: 244 bytes, Routine Base: \$CODE + 0000

; 201 0485 1
; 202 0486 1 End
; 203 0487 0 ELUDOM

.EXTRN LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes							
\$OWNS	136	NOVEC, WRT,	RD	NOEXE, NOSHR,	LCL,	REL,	CON,	PIC,	ALIGN(2)
\$PLIT	40	NOVEC, NOWRT,	RD	NOEXE, NOSHR,	LCL,	REL,	CON,	PIC,	ALIGN(2)
\$CODE	244	NOVEC, NOWRT,	RD	EXE, NOSHR,	LCL,	REL,	CON,	PIC,	ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	21	0	1000	00:01.9

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD, INITIAL, OPTIMIZE)/LIS=LIS\$:TRNS_BITS/OBJ=OBJ\$:TRNS_BITS MSRC\$:TRNS_BITS/UPDATE=(ENH\$:TRNS_BITS)
; Size: 244 code + 176 data bytes
; Run Time: 00:09.5


```

: Elapsed Time:      00:21.2
: Lines/CPU Min:    3079
: Lexemes/CPU-Min: 22501
: Memory Used:    133 pages
: Compilation Complete

```

TS

[illegible]

0154

AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY